# Illinois Web Accessibility Standards

Version 1.2
February 14, 2002

# Preface

In an effort to address the needs of all users, the Illinois Technology Office has established the Illinois Web Accessibility Standards (IWAS). On February 14, 2002, Governor George H. Ryan signed an Administrative Order directing Illinois agencies to "utilize the Illinois Web Accessibility Standards for the development of web sites, intranets, and web-based applications."

These standards are based on Federal "Section 508" and World Wide Web Consortium (W3C) accessibility guidelines, which were reviewed extensively by a panel of experts during the preparation of IWAS. You may wonder why the Illinois Technology Office has created an additional set of standards when the Section 508 and W3C standards already exist. The answer is quite simple: The federal Section 508 guidelines deal with a very basic level of accessibility needs while leaving out many issues facing users. The W3C guidelines, on the other hand, offer three tiers of priorities that go into greater depth than many government entities are able to address at this time. IWAS incorporate a combination of the two creating a standard well suited to serve the users of Illinois web sites.

# Contents:

# Introduction

## Purpose

The Illinois Web Accessibility Standards are designed to provide practical and specific guidance for the development of web sites, intranets, and web-based applications that are accessible to Illinoisans with disabilities.

## Audience

These standards are intended for use by all web authors, developers, and content contributors creating or maintaining web pages for the State. The basic concepts should be understandable by anyone with a general familiarity with web technologies. Knowledge of Hypertext Markup Language (HTML) and related web languages will help in fully understanding the Implementation Guidelines.

## Scope

These standards include:

1. Performance Criteria – a concise list of functional goals that must be achieved for a web site to be considered accessible.
2. Implementation Guidelines – the detailed techniques for addressing accessibility using current web technologies.

Both Performance Criteria and Implementation Guidelines are necessarily technology-dependent and will be updated as technologies evolve and change. The web technologies considered the current standards as of this version include:

- Hypertext Markup Language (HTML) 4.01
- Extensible Hypertext Markup Language (XHTML) 1.0
- Cascading Style Sheet (CSS) Level 1 & 2
- Document Object Model (DOM) Level 1
- Synchronized Multimedia Integration Language (SMIL) 1.0
- JavaScript & Dynamic HTML (DHTML)

Note: The use of other technologies (e.g., Java, Flash) and other document formats (e.g., Adobe Acrobat PDF, Microsoft Word, WordPerfect) is permissible if used in accordance with the standards outlined in this document. See the sections on Applets and Plug-ins and Downloadable Documents for more information.

## Relation to Existing Accessibility Standards

The Illinois Web Accessibility Standards build on two sets of existing standards:

- Federal "Section 508" Electronic and Information Technology Accessibility Standards for Web-based Intranet and Internet Information and Applications (http://www.section508.gov/index.cfm?FuseAction=Content&ID=12#Web)
- World Wide Web Consortium (W3C) Web Content Accessibility Guidelines (WCAG) 1.0 (http://www.w3.org/TR/WAI-WEBCONTENT)

The Illinois Web Accessibility Standards are designed to meet or exceed all Federal Section 508 requirements and all WCAG "Priority 1" Checkpoints. The Illinois Standards exceed the minimum requirements in many areas, incorporating a number of WCAG "Priority 2" and "Priority 3" Checkpoints as well as additional requirements identified through practical experience working with Illinoisans with disabilities. Each Guideline in the Illinois Standards includes a reference to the corresponding Section 508 requirements and WCAG Checkpoints.

# Performance Criteria:

Individuals with disabilities use a variety of accessibility techniques and assistive technologies to access web-based information. From a practical standpoint, web sites must therefore be compliant and compatible with these accessibility tools in order to be accessible to people with disabilities. From this perspective, the following functional performance criteria can be used to judge whether accessibility is effectively achieved:

### Illinois Web Accessibility Performance Criteria

All information and functionality presented in a web site, intranet, or web-based applications shall be available in a manner that is:

- Compliant with browser and system font size and color settings
- Completely operable using keyboard only
- Completely operable using leading screen magnification software
- Completely operable using leading screen reading software
- Completely operable using leading speech recognition software
- Completely understandable without sound
- Completely understandable without color
- Clear and consistent
- Unlikely to trigger photosensitive seizures

# Implementation Guidelines:

## 1. Coding

**1.1 – Use valid, standard web programming code.**

What: The World Wide Web Consortium (W3C) sets and publishes standards for most web programming languages, including HTML 4.01, XHTML 1.0, CSS Level 1 & 2, DOM, and SMIL. Programming code is considered "valid" when it follows the all the rules and conventions specified in the published standards.

Why: Screen readers and other assistive technologies can more accurately interpret and interact with web pages that are built using valid, standard code. W3C languages are designed with accessibility in mind.

How: Indicate the programming language you are using by starting your code with a document type declaration such as:
`<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">`
Use the W3C HTML Validation Service (http://validator.w3.org) and W3C CSS Validation Service (http://jigsaw.w3.org/css-validator) to check your code. Refer to the World Wide Web Consortium site (http://www.w3.org) for full specifications and documentation.

Ref: WCAG 3.2, 11.1

**1.2 – Use appropriate markup to convey document structure.**

What: HTML includes markup (programming code) to identify the structural elements of a document. For example, the `<p>` element identifies a paragraph and `<h1>` identifies a first-level heading.

Why: Screen readers use structural elements to help make reading more efficient. For example, some screen readers can skip from heading to heading to allow the user to "skim" the document.

How: Identify section heading, paragraphs, lists, quotes, etc using the appropriate tags instead of relying on formatting commands to distinguish these elements. For example, use `<h1>` tags to identify top-level headings rather than simply applying font-size or bold formatting commands. Do not misuse structural elements for formatting effects, such as using `<h1>` to make text bold or `<blockquote>` to indent a paragraph that is not actually a quotation.

Ref: WCAG 3.5, 3.6, 3.7, 5.4

### 1.3 – Use style sheets for formatting whenever possible.

What: Cascading Style Sheets (CSS) is a formatting language designed to compliment HTML. While HTML is designed to identify a document's structure, CSS is intended for formatting and presentation.

Why: In general, users can most easily override formatting settings made using CSS. The use of CSS for formatting also tends to facilitate the proper use of HTML to identify document structure.

How: See the W3C's Cascading Style Sheets site (http://www.w3.org/Style/CSS/) for specifications, tutorials, and resources.
Note: Some older web browsers, notably Internet Explorer 3 and Netscape 4, have problematic support for CSS. Be sure to test pages using CSS in multiple browsers.

Ref: WCAG 3.3

## 2. Text

### 2.1 – Avoid using images to display text.

What: Web developers often use images of text to achieve a specific style, size, or special effect.

Why: Users with limited vision rely on the ability to enlarge text or choose enhanced color combinations. However, most web browsers cannot change the size and color of images.

How: Whenever possible, use actual text instead of images of text. Style sheets can be used to achieve specific sizes, colors, or effects. Text that requires exact formatting, such as logos, are appropriate exceptions.

Ref: WCAG 3.1

### 2.2 – Avoid using absolute sizes for fonts.

What: Font sizes can be set using "absolute" or "relative" units of measurement. Absolute units, notably pixels, points, and inches, are based on fixed physical measurements; "relative" units, such as percentages or "`small`," "`medium`," or "`large`," are based on the user's default font size.

Why: Users with limited vision often rely on the ability to enlarge text. Most web browsers allow users to easily change the size of text that has been set with relative units (or not set at all). Using absolute font sizes

generally makes it much more difficult for users to change text size to meet their needs.

How: Set font sizes using relative measurements or avoid setting font sizes altogether.

Note: The "em" unit is another useful relative measurement; however, Internet Explorer 3 incorrectly interprets em's as pixels.

Ref: WCAG 3.4

### 2.3 – Specify the language of text.

What: HTML uses the `lang` attribute to specify language in a web page. It can be set for any HTML element.

Why: Words written in foreign languages can be unintelligible when spoken by a screen reader. Some screen readers are able to pronounce words in their appropriate language if it is specified.

How: Use the `lang` attribute on the `<html>` element to identify the primary language of each document, for example, `<html lang="en">`, for English.

Use the `lang` attribute on `<span>` or other elements to identify words or phrases in other languages. For example, a Spanish phrase within an English document could be coded as `<span lang="sp">se habla español</span>`.

Note: Not all screen readers support automatic language changes, but setting the `lang` attribute will not negatively affect those that do not.

Ref: WCAG 4.1, 4.3

### 2.4 – Avoid using "ASCII art."

What: "ASCII art" (and "emoticons") are images created using special arrangements of text characters and symbols. For example, ":-)" is often used to create a smiley face, and "->" is often used as an arrow.

Why: Screen readers read most ASCII art literally, which can be extremely confusing. For example, ":-)" reads as "colon dash right parenthesis," and "->" as "dash greater than."

How: Use images with appropriate alternate text instead of ASCII art.

Ref: n/a

## 3. Colors

**3.1 – Do not convey information with color alone.**

What: Color is often used to indicate special functions or status. For example, required form fields are frequently indicated with red labels.

Why: Users with blindness, limited vision, or color-blindness may miss information presented with color.

How: Whenever color is used as an indicator, use a non-color-based indicator as well. For example, required form fields could be identified with asterisks as well as color.

Ref: WCAG 2.1; 508 c

**3.2 – Use contrasting foreground and background colors.**

What: Web authors can set specific colors to be used for foregrounds (text) and backgrounds. Sometimes images are used as backgrounds.

Why: Users with limited vision or color-blindness may have difficulty reading text that is similar in color to its background.

How: For text, use dark colors on light backgrounds, or vice versa. Avoid combinations of red and green as well as busy background images.

Ref: WCAG 2.2

## 4. Images

**4.1 – Provide "alternate text" for all images.**

What: The HTML image element (`<img>`) includes an "alternate text" attribute (`alt`) that is used to provide text that can be substituted when the image itself cannot be displayed. Alternate text is meant to be a concise replacement for an image and should serve the same purpose and convey the same meaning.

Why: Individuals who are blind cannot perceive information presented in images; screen reading software reads alternate text instead.

How: ALL images must have appropriate alternate text. As a rule of thumb, consider what you might say if you were reading the web page to someone over the telephone. You do not need to include the words "image of" or "graphic of."

Specifically:

- For images that contain words or letters – use alternate text that includes the same words or letters.
- For images links – use alternate text that identifies the link's destination or function. You do not need to include the words "link to."
- For images that are invisible, purely decorative, or otherwise do not convey meaning – use `alt=""` (null) to indicate that the image can be safely ignored by a screen reader.

Ref: WCAG 1.1; 508 a

### 4.2 – Provide full descriptions for graphs, diagrams, and other meaningful images.

What: "Meaningful" images are images that convey more information than can appropriately be expressed as alternate text.

Why: A full description allows a user who cannot see or understand a meaningful image to receive the same information as a sighted individual.

How: Present a full description of a meaningful image either on the page on which the image appears or through a link immediately preceding or following the image. Use alternate text to provide a concise name for the image. For example, the alternate text of a graph should state its title and the long description should summarize its trends and/or present a table of its data.

Note: The `longdesc` attribute of the `<img>` element can also be used to provide a link to a full description. Because longdesc it is not yet supported by most web browsers, it should not be used as the only method of providing a full description.

Ref: WCAG 1.1; 508 a

## 5. Image Maps

### 5.1 – Provide alternate text for each area in client-side image maps.

What: Image maps are images divided into multiple "areas," with each area having its own hypertext link.

Why: Just as images must have alternate text, each area of an image map must also have appropriate alternate text for use when the image is not displayed.

How: Use alternate text that indicates the function or destination of the link for each area of a client-side image map. The image itself should have alternate text that indicates the overall function of the image map.

Ref: WCAG 1.1; 508 a

### 5.2 – Avoid using server-side image maps.

What: While client-side image maps and server-side image maps look and operate similarly, they are technically very different. Because of the way server-side image maps work, all information about the image and links is stored at the web server and is not available to the user's web browser or assistive technology.

Why: Screen readers cannot identify or read the separate areas or links within server-side image maps.

How: Whenever possible, use client-side image maps instead of server-side image maps. If server-side image maps must be used, provide a set of text links that duplicate all the functions/destinations included in the image map.

Ref: WCAG 1.2, 9.1; 508 e, f

## 6. Audio

### 6.1 – Do not convey information with sound alone.

What: It is possible to use sound for a variety of purposes, including presenting warning signals, cues, or verbal instructions.

Why: Users who are deaf or hard of hearing may miss information provided only through sound.

How: Whenever significant information is provided by sound, include a visual indicator that provides the same information as well.

Ref: WCAG 1.1; 508 a

### 6.2 – Provide text transcripts for audio containing speech.

What: "Audio containing speech" includes audio recordings or live broadcasts of speeches, seminars, conferences, etc. A text transcript is a word-for-word written record of the spoken content of such an event.

Why: Individuals who are deaf or hard of hearing may require text transcripts to access audio information.

How: Provide a link to a text (or HTML) transcript of any audio presented on a web site. Transcripts should be posted at the same time the audio is

made available. Computer-Aided Real Time (CART) captioners can transcribe live events.

Ref: WCAG 1.1; 508 a

## 7. Multimedia

### 7.1 – Provide synchronized captions for multimedia containing speech.

What: Multimedia generally refers to recorded or live media containing both video and audio tracks. Captioning (as in "closed captioned") is essentially a text transcript of the audio synchronized with the audio/video tracks of the presentation.

Why: Individuals who are deaf or hard of hearing may require captions to access the audio information in multimedia.

How: Whenever possible, captions should be implemented using Synchronized Multimedia Integration Language (SMIL) to synchronize the display of text from a transcript with the video. As a less desirable alternative, captions can be added to a standard video recording and then converted to a web format.

Ref: WCAG 1.4, 508 b

### 7.2 – Provide audio descriptions for multimedia with significant video.

What: Audio descriptions are verbal descriptions of the actions and images displayed in a video that are inserted during pauses in the regular dialog or audio track. Audio descriptions are only necessary if significant information that is presented visually is not discernable from the dialog or audio track.

Why: Individuals who are blind or low-vision may require audio descriptions to access the visual information in multimedia.

How: Carefully consider whether audio descriptions are necessary to present the significant information of a multimedia recording. Many speech-intensive events, such as speeches, lectures, or conferences, may not need audio description.

Ref: WCAG 1.3

## 8. Animation

**8.1 – Avoid flickering, blinking, and unnecessary animation.**

What: Animated graphics, Flash, Java, `<blink>` tags, `<marquee>` tags, and other techniques are often used to create a variety of animated effects.

Why: Flickering or blinking between 2 and 55 Hz (flashes per second) can trigger epileptic seizures. Animation can be distracting to users with certain visual or cognitive disabilities.

How: Do not cause elements to blink regularly between 2 and 55 Hz. Avoid animation and movement unless it provides significant additional information.

Ref: WCAG 7.1, 7.2, 7.3; 508 j

## 9. Links

**9.1 – Make sure that links are understandable out of context.**

What: A link is understandable out of context when it clearly indicates its destination or function without requiring additional information.

Why: Screen reader users often tab through links (skip from link to link by pressing the Tab key) in order to "scan" a page. Most screen readers also offer a "links list" feature to help speed the process of navigating to specific links. Links that are not understandable out of context, such as "click here" or "more," make these techniques much less efficient.

How: Use link text that is clear and unambiguous. Avoid using "click here."

Ref: WCAG 13.1

**9.2 – Provide a means of skipping past repetitive navigation links.**

What: Navigation links are the lists or "menus" of links to all the sections of a site that are often repeated on every page.

Why: Because navigation links are typically placed at the beginning (top left) of pages, screen reader users must read through all the navigation links before reaching the main area of the page. Individuals who use a keyboard instead of a mouse similarly must tab through all the navigation links before reaching the main area of the page. Providing a means of skipping these links can significantly improve efficiency and usability for screen reader and keyboard users.

How: Provide a link at the beginning of navigation lists that points to a target at the beginning of the main content area of the page. This link must be visible to screen reader and keyboard users, but can be hidden from other users. It is also acceptable to design a page so that navigation links come at the end of the document.

Ref: 508 o

### 9.3 – Avoid using small images or text as links.

What: The size of the "clickable" area of a link is limited to the size of the image or text that makes up the link.

Why: Mouse-users with limited fine motor control may have difficulty pointing to and clicking on links that are small, especially if the links are close together.

How: Make sure that images used for links are reasonably large, preferably 32 pixels by 16 pixels or larger. Use standard or enlarged font sizes for text links, and avoid using text links that are shorter than four characters in length. Additionally, avoid placing small links close together.

Ref: n/a

## 10. Forms

### 10.1 – Associate labels with all form fields.

What: HTML forms include "fields" such as buttons (`<input type="button">`), text boxes (`<input type="text">`), list boxes (`<select>`), and more. Each field is typically identified by a text label.

Why: Screen readers cannot always determine which label belongs to which field based on positioning alone. The `<label>` element makes this association clear.

How: Use the `<label for="">` tag to label every form field.

Note: The value of a label's `for` attribute is the corresponding field's `id`, not its `name`.

Ref: WCAG 12.4; 508 n

### 10.2 – Position labels as close as possible to form fields.

What: Using certain layout techniques, form labels are not always positioned immediately next to their fields.

Why: When screen magnification software enlarges a web page it also reduces the field of view. If form field label is positioned far away from its field, it may be impossible for a screen magnifier-user to view both the field and the label at the same time.

How: Position labels immediately adjacent to fields, preferably in standard locations, such as on the left or above text boxes and list boxes and on the right of checkboxes and radio buttons.

Ref: WCAG 10.2; 508 n

**10.3 – Include any special instructions within field labels.**

What: Frequently, special instructions are listed after the field to which they apply. In some cases, instructions are even presented in "pop up" text or in the browser's status bar.

Why: When filling out a form, screen readers typically read only the field's label. Screen magnifiers will focus on the field and its label, and instructions may be out of the field of view.

How: Special instructions should be given before the form field and within the field label if possible. If instructions are too long to appropriately fit within the label, they should be given in an instructions section in advance of the form.

Ref: 508 n

**10.4 – Make sure that form fields are in a logical tab order.**

What: Screen reader and keyboard users move between form fields (and links) using the Tab key. The order in which form fields receive focus is called the tab order. By default, the tab order follows the order in which elements appear in a page's HTML code.

Why: Depending on the design and layout of a page, the tab order may not match the visual (or logical) order of fields on a form. Reading fields out of their intended order can be disorienting for a screen reader or keyboard user.

How: Make sure that fields appear in the HTML code in the logical order and/or use tabindex to set the appropriate order.

Note: `Tabindex` only is supported by Internet Explorer 4 and up.

Ref: WCAG 9.4; 508 n

## 11. Data Tables

**11.1 – For simple data tables, explicitly identify headings for all columns and rows.**

What: "Data tables" are simply HTML tables used to display data. (On the other hand, "layout tables" are used to lay out columns and sections on a web page. Both data and layout tables use the `<table>` element, but their functions, and accessibility issues, are very different.) "Headers" identify the content of each row and/or column.

Why: A screen reader can use table headers to provide row and column information while a user explores the data cells within a table.

How: Use `<th>` (table header) or `<td>` (table data) elements with `scope="col"` (for column headers) or `scope="row"` (for row headers) to identify cells that contain row and/or column headings.

Ref: WCAG 5.1; 508 g

**11.2 – Avoid using complex data tables.**

What: Table with multiple layers of headers and "spanned" columns or rows can become very complex.

Why: Complex data tables can be difficult to navigate and understand using a screen reader. Only the most advanced screen readers can use advanced table markup to provide orientation information.

How: Whenever possible, simplify complex tables by re-arranging or dividing them into separate tables. When a complex table cannot be simplified, use advanced table markup, such as headers, axis, scope, `<col>`, and `<colgroup>`, to fully indicate the relationships between data cells and headers.

Note: See W3C's "Tables in HTML Documents" (http://www.w3.org/TR/html401/struct/tables.html) for complete details on how to markup complex tables.

Ref: WCAG 5.2; 508 h

## 12. Frames

**12.1 – Provide meaningful names and page titles for all frames.**

What: HTML frames are used to divide web pages into separate areas, each displaying a separate web page. Each frame is identified by a `name`

attribute and each page contained within a frame is identified by its `<title>` element.

Why: To navigate pages with frames, users who are blind must be able to identify the different frames and understand the purpose of each frame. Most screen readers identify frames by speaking the name and/or page title of each frame.

How: Give each frame an understandable name that indicates the frame's function. For example, use `name="Navigation"` and `name="Content"` rather than `name="nav"` and `name="right"`. Set the `<title>` element of each page contained within a frame to match the `name` attributes or to identify the current content of that frame.

Note: Traditionally, the "`name`" attribute is used for programming and should not contain spaces; the `title` attribute, which can contain spaces, can also be used to set a more descriptive name for each frame; however, this technique is not yet supported by all screen readers.

Ref: WCAG 12.1; 508 i

### 12.2 – Avoid using empty or non-essential frames.

What: Frames are sometimes used inappropriately for formatting and layout. For example, empty frames can be used to create margins around or within a page.

Why: Screen readers cannot judge whether the content of a frame is significant and must identify every frame for the user. Having to read this extraneous information for non-essential frames can be time consuming and confusing.

How: Use frames sparingly. If a frame is not necessary for page content, eliminate it.

Ref: n/a

## 13. Scripts

### 13.1 – Make sure that significant interactions can be performed with both keyboard and mouse.

What: Scripting languages, such as JavaScript, are simple programming languages that can be used within a web browser to automate certain tasks and enable pages to change and respond to user input. Scripts can trigger changes when the user performs specific actions ("events"). Some events are triggered by either mouse or keyboard

actions. For example, an image can change color when the mouse pointer hovers over it (the onmouseover event).

Why: Users with physical impairments may be able to use the keyboard but not the mouse. Individuals who cannot see the mouse pointer on the screen also use the keyboard for all interactions. Scripts that can only be triggered by the mouse are not usable by these individuals.

How: Whenever using a mouse-only event (e.g., onmouseover, onmouseout) to trigger a significant script action, also use the corresponding keyboard event (e.g., onfocus, onblur). Also make sure that keyboard events do not unintentionally trigger script actions. For example, keyboard users should be able to arrow through the choices in a `<select>` list without triggering each choice (e.g., onchange).

Ref: WCAG 6.4, 9.2, 9.3

### 13.2 – Make sure that essential content and functionality are available when client-side scripts are not fully supported.

What: Scripts are often used to dynamically show or hide the content that appears on a web page or to perform important functions, such as checking that entries in form fields are appropriate. "Client-side" scripts, such as JavaScript, are scripts that are run by the user's web browser. Client-side scripts must be supported by and compatible with the user's browser in order to work. ("Server-side" scripts, such as CGI, ASP, JSP, or PHP, run on the web server before the web page ever reaches the user's browser. Server-side scripts do not generally pose additional accessibility problems.)

Why: Older assistive technologies and web browsers may not support client-side scripting at all. Even current assistive technologies may interact in unexpected ways with content that is displayed using scripts, such as by skipping text that is dynamically displayed or reading text that is dynamically hidden. Users need to be able to access the same essential content and functionality whether scripts are fully, partially, or not supported. It is not safe to assume that users with disabilities will have scripting support turned off.

How: Whenever scripts are used, it is the responsibility of the page developer to thoroughly test using assistive technologies to ensure that all information and functionality is accessible. If there is any doubt, err on the safe side by ensuring that the essential elements of the page do not rely on scripts.

Note: One approach to ensuring accessibility with scripts is to include a back-up method of providing the same information and functionality that does not require scripts. For example, if a client-side script is used

to check an entry in a form field, a server-side script could make the same check. Similarly, if scripts are used for "drop-down" menus, the same menu choices could be provided in an appropriate location elsewhere on the current or subsequent page. Additionally, scripting features that are purely decorative and do not present any significant information or functionality do not need to be made accessible. (However, remember Guideline 8.1 – Avoid flickering, blinking, and unnecessary animation.)

Ref: WCAG 6.3; 508 l

## 14. Applets and Plug-Ins

**14.1 – Use accessible applets or plug-ins whenever possible.**

What: "Applets" and "plug-ins" refer to a variety of newer web technologies, such as Java and Flash, that can be used to create advanced, interactive content on web pages. Both require additional software to be downloaded, installed, and run before the content can be viewed or used. Applets and plug-ins also operate with their own user interfaces, which are separate and different from that of standard web pages.

Why: Because applets and plug-ins have their own interfaces, they must be accessible in and of themselves. If essential content or functionality is presented using an applet or plug-in that is not accessible, it will not be usable by individuals with disabilities.

How: Check with the manufacturer and/or developer of each applet or plug-in to determine if and how the technology is accessible. When an accessible applet or plug-in is available, provide users with a link to any special instructions or software that is necessary.

Ref: WCAG 8.1; 508 m

**14.2 – If an inaccessible applet or plug-in must be used, provide an accessible alternative that includes the same content and functionality.**

What: If an applet or plug-in is inaccessible, it may be possible to provide both the original applet or object and an equivalent accessible alternative.

Why: The same features that make an applet or plug-in inaccessible to some users may actually improve accessibility or usability for users without, or with different, disabilities. By providing both the original and accessible versions, the same content and functionality can be available to all users.

How: Wherever a link is provided to an inaccessible applet or object, also provide a link to an equivalent accessible version. Make sure that the information and functionality is completely equivalent and up-to-date. Be sure to consider whether the inaccessible version is actually necessary.
In cases where it is impossible to create an equivalent accessible version, such as with some geographical imaging and mapping systems, exceptions may be necessary.

Ref: WCAG 6.2, 11.4; 508 k

## 15. Downloadable Documents

**15.1 – Provide accessible HTML or text versions of downloadable documents whenever possible.**

What: Downloadable documents are often provided in formats such as Adobe Acrobat PDF, Microsoft Word, or WordPerfect. Such documents must be viewed in their own applications or using a web browser plug-in.

Why: The applications required to open downloadable documents may not be available or accessible to users with disabilities.

How: Wherever a link is provided to a document that is not HTML or text, also provide a link to an accessible HTML or text version of the same document. HTML versions should follow these guidelines; text versions may require reformatting to ensure proper reading order, and additional text descriptions may need to be added for charts, graphs, or other non-text content.

Note: Adobe is actively improving the accessibility of PDF documents, however, the process for making existing PDF documents accessible is complex, and the accessibility features are not yet completely supported. (See access.adobe.com for more information as well as Adobe's online PDF to HTML conversion tools.)

Ref: n/a

**15.2 – If a downloadable document cannot be provided in an accessible electronic format, provide information on how to request an alternate format.**

What: In some cases, documents cannot be provided in electronic format.

Why: Users with disabilities must still have equivalent access to public documents.

How: Provide information regarding whom to contact to obtain the document in alternate formats (e.g., braille, large-print, or audio-cassette). Alternate formats must be available in a timely manner.

Ref: n/a

## 16. Window Control

### 16.1 – Notify users of actions that will open a new window.

What: It is possible to cause hypertext links to open pages in a new browser window, or to automatically open additional windows when a page loads or unloads.

Why: It may not always be obvious to users, especially those with limited vision, blindness, or cognitive disabilities, when a new window has opened. It can be confusing when features such as the browser's "back" button no longer work as expected.

How: Avoid automatically opening new windows. Clearly identify any links that will open new windows by providing an indication in the link text or `title` attributes.

Ref: WCAG 10.1

### 16.2 – Do not automatically refresh the current page.

What: It is possible to cause web pages to automatically re-load their content on a certain interval. For example, a page containing news headlines might refresh every few minutes to present the most current items.

Why: When a page automatically refreshes, it can cause a screen reader to re-start reading from the beginning of the page.

How: Do not use `HTTP-EQUIV="refresh"`. If necessary, provide a link or control to allow the user to refresh a page at his or her discretion.

Ref: WCAG 7.4

### 16.3 – Notify users of time limits and provide a means to extend time if needed.

What: Some web pages, frequently those that require a user to log in with an ID and password, "reset" themselves after a certain period of inactivity. Typically, any form entries that have been partially completed are erased and the user must start over.

Why: Users with visual, physical, or cognitive disabilities may require more time than average to read and interact with a web page.

How: Provide a clear explanation of any time limits and offer the user a way to extend or remove the limits if necessary. Avoid using time limits unnecessarily.

Ref: WCAG 7.5; 508 p

## 17. Page Layout

### 17.1 – When using tables for layout, make sure that reading order is logical.

What: Layout tables are HTML tables used to lay out a web page in multiple columns and sections (as opposed to tables that actually present data.) "Reading order" refers to the order in which a screen reader would read through the table. For example, the reading order for a simple table might be (1) row 1, cell 1, (2) row 1, cell 2, (3) row 2, cell 1, and (4) row 2, cell 2.

Why: Screen readers read through tables in the order in which cells are defined in the table code, which can be very different from the order that someone reading visually would follow. It is essential that the reading order match the logical flow of the document so that a screen reader user would hear the document in the same order that a visual reader would read it.

How: Check the reading order by following the order in which the table cells appear in the code. It may be possible to combine cells and/or nest tables to achieve an appropriate reading order.

Ref: WCAG 5.3

### 17.2 – When using style sheets for layout, make sure that reading order is logical when style sheets are not supported.

What: The positioning features of Cascading Style Sheets can be used to position elements visually almost anywhere on a web page.

Why: As with layout tables, screen readers read through the elements on a web page in the order in which they appear in the page code, regardless of how they are positioned using style sheets. It is essential that the reading order match the logical flow of the document so that a screen reader user would hear the document in the same order that a visual reader would read it.

How: Check the reading order by following the order in which elements appear in the page code. Reading order can usually be adjusted by rearranging the order in which elements are defined in the code.

Ref: WCAG 6.1; 508 d

### 17.3 – Minimize the need for horizontal scrolling.

What: If a web page is wider than the window or screen in which it is viewed, most browsers will display a horizontal scroll bar and require the user to manually scroll to see the entire page.

Why: When a screen magnifier enlarges a web page, it also reduces the field of view so that the user must pan (scroll) to see the entire page. When the web page being viewed also requires horizontal scrolling, the combination can be awkward or unusable. Keyboard users may also find repetitive scrolling fatiguing and inefficient.

How: Design pages so that they can resize to fit the width of the user's browser. Use relative widths on tables and frames used for layout and make sure that horizontally adjacent images are less than a total of 600 pixels wide. If scrolling cannot be avoided, place the least important content in the rightmost part of the page.

Ref: WCAG 3.4

## 18. Page Content

### 18.1 – Use the clearest, simplest, and most concise language appropriate for a page's subject matter.

What: "Clearest, simplest, and most concise language" refers to the words and grammar used in the content of a web page. It is a subjective goal that depends on the subject matter and intended audience of each web page.

Why: Clear and simple language is easier for all readers, and especially those with cognitive or learning disabilities. Simple language also helps individuals whose primary language is American Sign Language, which differs significantly from written English.

How: Be concise and avoid jargon. Have someone else proofread your text. Do user testing with people from your intended audience if possible.

Ref: WCAG 14.1

## 19. Alternate Accessible Versions

### 19.1 – Use separate accessible versions only as a last resort.

What: Separate accessible or "text-only" versions are often offered instead of providing a single accessible site.

Why: Manually developing and maintaining a separate "text-only" version of an entire site is tremendously demanding of time and resources. In practice, "text-only" versions are rarely kept complete or up-to-date. Given advances in accessibility techniques and assistive technologies, "text-only" sites are simply not necessary in most cases.

How: Follow these standards to develop a single site that is universally accessible and efficient to maintain.

Ref: WCAG 11.4; 508 k

## 20. Contact Information

### 20.1 – Provide contact information.

What: A contact person for accessibility issues should be identified. Contact information should include email, telephone, TTY, and mailing address.

Why: Individuals with disabilities may need to report accessibility problems or request information in an alternate accessible format.

How: List accessibility contact information on the home page or contact page. Inquiries about accessibility, especially requests for materials in alternate format, need to be handled in a timely manner. This contact information should also be provided to the Illinois Technology Office (see Credits & Contacts below).

Ref: n/a

## 21. Testing

### 21.1 – Test for accessibility.

What: Testing includes functional tests with assistive technology, browser and operating system functionality as well as automated testing software.

Why: Testing will determine whether accessibility has actually been accomplished.

How: Use browser and operating system accessibility features and leading assistive technology software such as screen readers and magnifiers to test for functional accessibility. Use an automated testing tool like Bobby (http://www.cast.org/bobby) to identify common accessibility problems. If possible, do user testing with people with disabilities.

Ref: n/a

# Credits & Contacts


Author: Mike Scott, MSF&W Information Technology Solutions

Contributors: Patrick Beaird, Illinois Technology Office

Jon Gunderson, University of Illinois at Urbana-Champaign, Center for Instructional Technology Accessibility

Wilhelmina Gunther, Illinois Assistive Technology Project


For questions, comments, or suggestions regarding the Illinois Web Accessibility Standards, contact the Illinois Technology Office:


Illinois Technology Office
2½ State House
Springfield, Illinois 62706
Telephone: (217) 557-5944
TDD 217-782-2239□□
Email: webmaster@state.il.us
Web: http://www.state.il.us/tech/technology/


For the most current version of these standards, please see:

http://www.state.il.us/tech/technology/accessibility